The Journal of Animal & Plant Sciences, 25(4): 2015, Page: 1176-1182
ISSN: 1018-7081

TOWARDS A SECURED DESIGN METHODOLOGY FOR CLOUD BASED BIOINFORMATICS APPLICATIONS

A. Nasir and M. U. G. Khan

Department of Computer Science and Engineering, Bio-informatics Lab, Al-Khwarizmi Institute of computer Sciences
University of Engineering and Technology, Lahore, Pakistan
Corresponding author: nasir_bhutta@yahoo.com

ABSTRACT

The amount of molecular biology data currently being generated from the lab based testing and conventional computing techniques was not thought before. But at the same time the existing techniques remains incapable to cater this massive amount data sets to retrieve meaningful results in an efficient manner. With the exponential growth of computing applications and their corresponding users, many evolutionary systems have appeared in the current markets, such as cloud computing, grid computing, bioinformatics, and video surveillance systems. Though the discovery of such systems has brought fruitful changes in daily lives of their users, there are several severe problems associated with such computing environments and security is one of the most critical issues faced by the users and service providers of these applications. These security issues and problems may arise during the life span of any evolutionary domain i.e. they are dynamic in nature as they arise and need to be handled during their life span. In this paper, we have introduced a secured methodology for designing evolutionary computing applications. We further present a working system from bioinformatics domain for testing and evaluation of the proposed design methodology.

Keywords: Cloud Computing, Bioinformatics, Evolutionary System, Secured Design Methodology, Software Engineering, Software Designs.

INTRODUCTION

Since the inception of molecular biology, huge amount of data is being generated from the lab based testing. Conventional computing techniques remain generally incapable to cater massive amount of meaningful and useable data sets. In the recent past cloud-based bioinformatics has facilitated the molecular biologists working in the areas of agriculture, livestock and biomedical sciences to tackle enormous amount of data and to make more useful applications. Moreover, it has been helping the research community of the life sciences to develop such applications to diagnose the life threating diseases and their surveillance.

With the introduction of new computing domains such as bioinformatics and image processing in the cloud paradigm, several potential problems have appeared for cloud computing developers. For these domains, which are also called evolutionary domains, the data types and functional requirements are dynamic in nature. The peculiar characteristics of evolutionary domains differentiate them from conventional applications due to the ever changing user requirements and security issues. Therefore existing design methodologies are unsuitable to use for the development of evolutionary applications. One of the major reasons for their unsuitability is their dynamicity and security concerns.

The main advantages of cloud computing dimension for bioinformatics (Kim *et al*., 2012 and Rosenthal *et al*., 2009) are as follows:

i. Biological data is evolutionary which is increasing exponentially; storage of this data on temporal basis will be cost reductive.
ii. Globalize this uniform biological data for user utilization via an internet connection provides an extra flavor of mobility.
iii. Provides 24/7 access to the uniform biological data.
iv. Provide more computation power.
v. Economically is suitable with respect to hardware and software cost.

To address such issues we propose a design methodology for incorporating the ever changing requirements of the evolutionary domains. Through this, we intend to introduce security and privacy throughout all phases of the classical waterfall model of software development. Because security risk is not fixed, therefore the security requirements are dynamic due to internal and external vulnerabilities and require improvements in response to new threats (Khan *et al*., 2015).

The objective is how to meet the security and privacy challenges that arise from broad spectrum such as internet etc. The design of bioinformatics software applications for cloud paradigm is taken as a running example, which can be extended further for any other evolutionary system.

As biological data is increasing exponentially, so there is a need to store and understand this data using computational techniques. Bioinformatics is an application that deals with the storage and interpretation of biological data. Main features (Luscombe *et al*., 2001) that are provided by bioinformatics are to:

i. Provide the optimal ways to exploit the existing biological data and to enhance this data;
ii. Develop such tools by which the analysis of biological data can be performed and
iii. Utilize these tools to interpret the results in biological manner.

There are numerous problems that bioinformatics domain is facing such as the existing biological data which is electronically available has heterogeneous nature (Khan *et al*., 2015) e.g. different formats of storage, which is the main hindrance of optimal utilization of biological data, integration of various biological data sets and retrieval of the required facts from bulk of data with speed (Goble *et al*., 2008, Min-Huang *et al*., 2003).

The main difference between the development processes of the conventional applications and cloud applications is that the cloud applications are developed on assumptions and requirements of service providers, and they are gathered from the market surveys (Lynch, 2009). The task of collection of the requirements in conventional application is usually one time task during the development. Whereas, in the cloud computing environments security requirements may even change during the life span of a cloud computing application.

To the best of our knowledge, the existing software development methodologies being used for cloud also do not have any iterative analysis and design process which caters the changing requirements of user and their feedback. We continue discussion of our proposed work in the next section. Classical water fall life cycle model provides a framework for software design methodologies, but it does not provide explicit guidelines (or instructions) how to handle evolutionary type of changes (such as security requirements of users, service providers, or both) that occur during development and after the development of cloud applications.

MATERIALS AND METHODS

This methodology is proposed keeping in view main concern of the users of Cloud applications i.e., security, because millions of users are interacting through web and their security requirements are dynamic in nature. This design methodology consists of two (2) phases, i.e., Analysis Phase and Design Phase. We have introduced back cycle (BC-2) within the Analysis Phase of Waterfall model. The proposed methodology also introduces the concept of Monitoring tables (RMT, SRT), which are used during development and post deployment.

Analysis Phase: This proposed life cycle model consists of three sub phases (SP), i.e., *Requirement Gathering* (SP-I), *Refining of Requirements* (SP-II) and *Processing of Requirement Analysis Algebra* (SP-III) as shown in Figure 1. Briefly depiction of these sub phases is given as under:

***SP-I*: Requirements Gathering**

i. The input of *SP-I* is Problem statement.
ii. Functional requirements and implicit constraints are gathered in raw form. Security requirements are also gathered during SP-I keeping in view standards and technology in consultation of users etc.
iii. Output of *SP-I* is F_{raw}(functional requirements in raw form), C_{raw} (Constraints in raw form) and *S* (Security Requirements) along with User and their Tiers (w.r.t. responsibility of different users in the organization).

SP-II: Refinements of Requirements

i. This sub phase focuses on refinements of F_{raw}, C_{raw}. The refinement of association of users into their respective tiers is performed at this stage.
ii. The output of *Refinement* sub phase is Functional requirements, Constraints and user along with their respective tier's information. Whereas set S is not part of refinement process.

SP-III: Processing of Algebra (RAA)

i. The *Need* operator is processed on the set *F* to associate security requirements with elements of set *F*.
ii. The *Dependency* operator is processed among elements of set *F*, to determine dependency among them.
iii. Such security requirements are excluded because of applying the dependency operator.

***SP-I*: Requirements Gathering:** In this sub-phase, all types of requirements and other information (related to user's organization, input, output, data files) are gathered.

Step I: The organization and its structure is studied with consultation of employees and Problem Statement. This includes its working environment, flow of information from one point to another point, function and responsibilities of each employee/user of the organization. This step identifies two sets (list) and their inter relationship such as:

i. Set of users/employees of the organization; user's details of the client organizations are provided.
ii. Set of tiers of the organization; details of different tiers (organizational structure w.r.t. responsibility e.g., *Top* management) of the client organizational are listed.
iii. A tentative relationship among users and their corresponding tiers is established.

iv. A tentative list of the users with their respective tier in the organizational structure is prepared, such as $\{T_1, T_2, T_3 \ldots Tp\}$ is the set of tires and $\{U_1, U_2, U_3 \ldots Uq\}$ is the set of users.

Step II: Identify and enlist the following three (3) sets, i.e., $\{F_{raw}, C_{raw}, S\}$, such that:

i. Set of functional requirements in raw form (F_{raw}): Functional requirements illustrate what the system should do by identifying the necessary task, action or activity that must be its part.

a) Input and Output of each function.

b) Data files, in case of their use for input/output.

ii. Set of security requirements (S)

iii. Set of constraints in raw form (C_{raw})

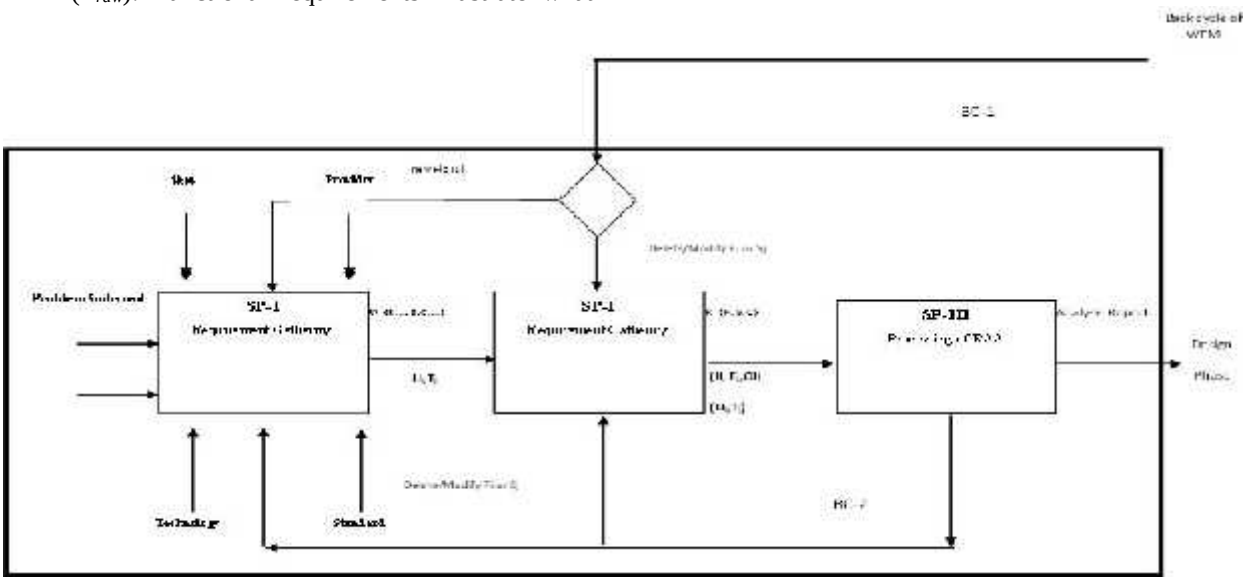

Figure 1: Analysis Phase of the proposed Methodology.

Realization of Monitoring Tables: During this sub phase two monitoring tables are realized, i.e., Requirement Monitoring Table (RMT) and Security Repository Table (SRT). These tables are dynamic in nature, because they are frequently updated when some predefined events occur in the system.

RMT(Function, Dependency, Security, User and Enable) (I)

SRT(Security, Category, Ranking and Enable) (II)

In expression (I), *RMT.Function* represents a list of functional requirements. *RMT.Dependency* maintains dependency among functional requirements (if exists), between the members of the set F by applying operator $\check{D}$ and *RMT.Security* points out security requirements associated with different functional requirements. The attribute *RMT.User* shows the list of users availing the corresponding functional requirements and will pop-up when the system is in operation. Whereas, *RMT.Enable* indicates whether the function is enabled or disabled by the Administrator.

SP-II: Refining of Requirements: The function of this sub-phase, *SP II*, is to refine the set of functional requirements, implicit constraints of an application and the list of user and their respective tiers. Following rules are defined for refining Functional requirements:

Rule-I: If (Inputs of any two functional requirements are same) ^ (Outputs of the two functional requirements are same) ^ (their names are same).

Rule-II: If (Inputs of any two functional requirements are same) ^ (Outputs of the two functional requirements are same) ^ (their names are different).

Rule-III: If (Inputs of any two functional requirements are same) ^ (Outputs of the two functional requirements are same) ^ (their names are different).

Rule-IV: If (Inputs of any two functional requirements are same) ^ (Outputs of one of these functional requirements are subset

Rule-V: If (Inputs of one of the functional requirements are subset of input of other) ^ (Outputs of both functional requirements are same).

Procedure

Step-1: For each element, C_k, of the set C_{raw}, identify all the elements in the set C_{raw} that are the same to the element, C_k, with the help of experts,

Step-2: Delete all same elements from the set C_{raw}, that are marked as the same to the element c_K,
Step-3: Put remaining elements in the set C;
End of procedure

Figure 2: Procedure of Refining Constraints

For refinement of user tiers, following procedure is applied.

Procedure: Refining_User_*Tiers*
Input= *Tentative list of Users and tiers*
Output= Refined list of Users and tiers in terms of 2-tuples i.e., (U_i,T_j)
Step-: i=1
while i≤ q do
/* for all users */
Step-2: For each user finalize his tier in consultation of organization authority.
Step-3: Develop relationship in terms of 2-tuples i.e., (U_i,T_j)
/* where i= 1 to p and j=1 to q */
end of while
end of procedure

Figure 3: Refining User and Tiers

SP-III: Processing of Requirements Analysis Algebra: *Requirement Analysis Algebra* (RAA) proposed in 2013 (Nasir, 2013) is applied to elements of the set *S*, which are associated with one or more elements of set *F*, are marked and placed in Requirement Monitoring table.

Step-I: Processing of N_d

Procedure: *Processing_*N_d
Input= (*F,S)*
/* Both sets *F* and security requirements *S*/
Step 1: i=0;
while i≤|F| do
/* For each element of the set *Fi* Call operator $N_{d\ i.e.\ !F!}$*/
{
Step 2: Call operator N_d for each F_i to associate corresponding security requirements to make it secure;
Step 3: i=i+1
}
end of while
end procedure

Figure 4: Procedure of *Processing_*N_d

Step-II: Processing of $\check{D}$

Step 1: i=0
while i≤|f| do
{
Step 2: Call *Operator* $\check{D}$ for each element of the set *F to identify dependency among functional requirement.*
/* If a functional requirement has pre-requisite(s) functional requirement(s), then the operator among them is referred as *dependent.*
Step 3: k=0
While K=|F_i*| do
/* Where F_i*⊂*F*, are those elements of the set *F* upon which F_i *is dependent*/*
Step 4: Exclude security requirements from F_i, which are applied on such functions upon which

Figure 5: Procedure of *Processing_*$\check{D}$

Design Phase: In design phase, the different access pattern of users for services and data are assigned secured (1) or unsecured (0) tags based on three rules provided in *Security Classification* module. Finally respective permissions are granted or revoked based on the returned security tags. To the extent of secured access the security requirements are ignored, resulting better system performance with respect to efficiency, whereas for unsecured services and data, we apply security requirements identified during Analysis Phase, to make them secure through proposed design phase discussed in the following.

In literature security mechanism is presented by three steps, *Authentication*, *Authorization* and *Services Allocation*. We extend this further for more two steps i.e. *Identification* and *Security Classification*. The *Design Phase* of the proposed methodology consists of following five modules:

i. *Identification;* This step checks availability of user in the system database. It can be referred as the procedure to restrict the access of application to only designated users. Because it is necessary to verify that a user claiming to be is registered for having access to the cloud application.
ii. *Authentication*; It is a step ahead of identification where identified user is further validated through other credentials such as password, biometric or smart card etc.
iii. *Security Classification;* A new module is proposed between authentication and authorization to handle respective user classes' access patterns. In current work only binary classification is proposed where these access patterns are in terms of either S*ecure* (1) or U*n_Secure*(0).
iv. *Authorization*; this module involves access allocation rights to an authenticated user. Because once a user is successfully identified and authenticated as genuine then it must be determined what data is permitted to access and what operation (services) are allowed to perform.
v. *Provision of Services;* this module enlists services (functions) that are provided by the cloud application to its clients.

Output of the Design phase is security and privacy design specification known as *Design Report* (DR). *Design Report* mainly consists of desired features and functions (services) in detail, including business rules, process diagrams, pseudo code and other documentation. These design elements are intended to describe the system in sufficient detail, which is input for next phase i.e. Implementation.

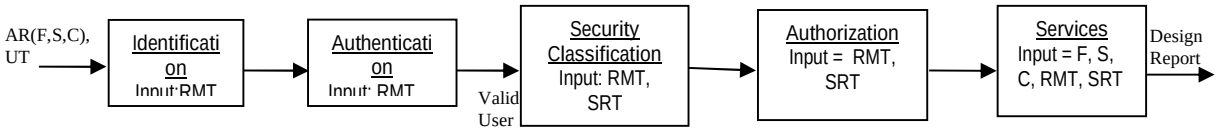

Figure 6: Proposed Design Phase

Identification module: Identifies whether a user is genuine or not by checking its existence in the user table i.e., *RMT* and if exists then he/she can proceed further to the subsequent module i.e. *Authentication*. In case a user is not registered that is does not exist in the user table (*RMT*), than access to further modules is denied unless a valid user account is created.

Authentication: Once the user is verified that he/she is genuine then he provides other credentials such as password or pin-code for authentication. Which is preformed either by service provider himself by using authentication database or outsourced to some external entity. However, outsourcing of authentication from external entities for smaller cloud deployment is not preferred (Almulla, *et al*., 2010).

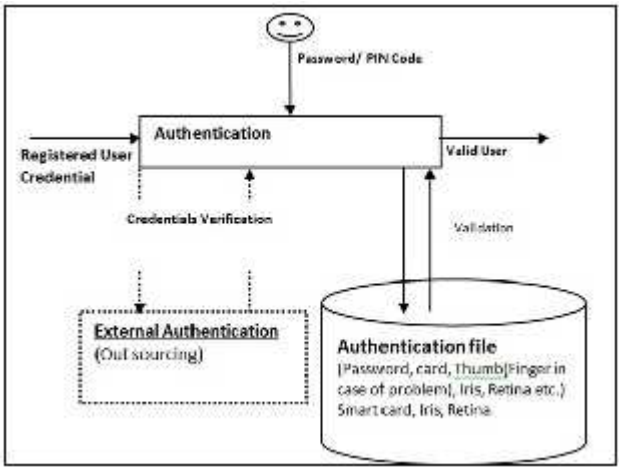

Figure 7. Procedure for Authentication.

Single level authentication can have several shortcomings such as weak passwords(can be guessed easily based on user's beliefs and historical background), errors while capturing biometric features like tilted cameras or thumb impression devices etc. To remedy these shortcomings, multi-level approach needs to be explored which combines several parameters. Therefore, we have used multi-level authentication by taking Cartesian product of different parameters used for the authentication.

Security Classification: Once the user is authenticated, then it is required to decide the services and data accessible to the legitimate users of the application. The classification of the three parameters (U, Serv, D) i.e., users, services and data provides guide line to sort out such problem in an efficient manner at early stage of the development process. For this purpose RMT and SRT are used for classification purposes. Based on following rules these parameters are assigned secured (1) or unsecured (0) tags to determine security for different user's access to services and data. Let Ui is a user, then. Finally authorization is performed where respective permission are granted or revoked based on the returned security tags from this module.

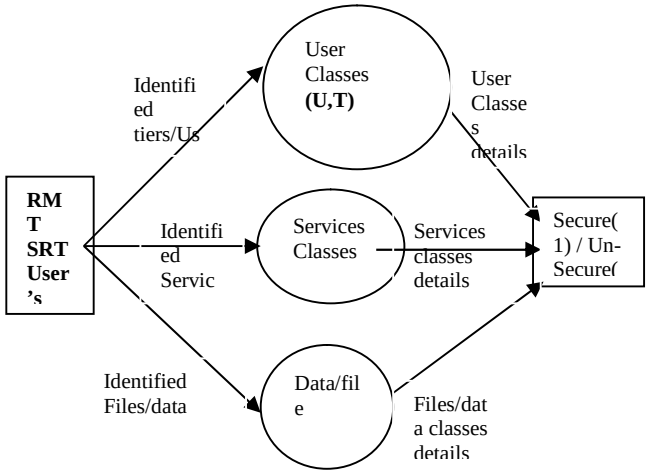

Figure 8: Decomposition of Classification Sub-Phase (Design Phase)

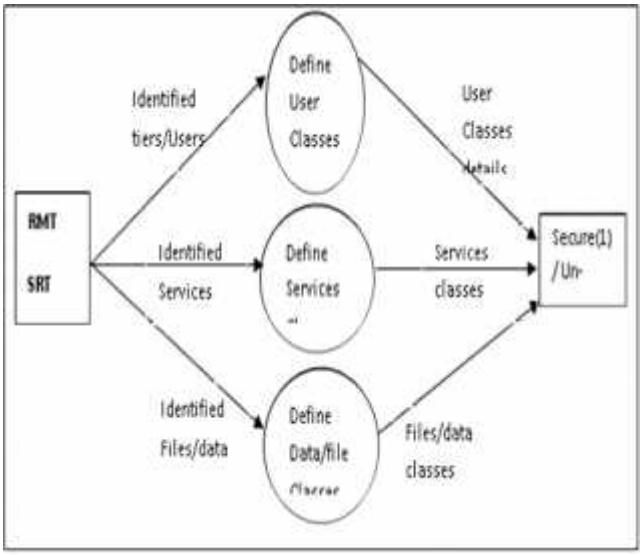

Figure 8. Decomposition of Classification Sub-Phase (Design Phase).

Authorization is the process of verifying whether an authenticated user has access to protected resources. Authorization is usually implemented near to the object which needs to be protected. The procedure of Authorization is as follows:

```
Procedure: Authorization
Step 1:  Check<authorize>
         Log-in access
         otherwise message "not authorize"
/* currently not authorize*/
Step 2:  Examine user-class
Step 3: execution services
/* operation according to user-class specific*/
Step 4:  Allow data set
         Select data-set
/* data set for specific user*/
End of procedure
```

Figure 9: Authorization procedure.

Design of Services (Functional requirements): Security of software applications depends upon its services which are provided by the cloud application to the different users. This varies from service to service. For example, security levels of various services (functions) of a Banking Application's are different such as ATM module security will be more critical as compare to opening of account.

```
Procedure: Design-Services
Input= AR{F,S,C}
Output= Services design
Step 1: i=1
while i≤ n do
/*|F|=n, where Fi∈{F1,F2,F3…Fn} */
{
Step 2: Design service along with Input and Output
Step 3: Whether the service Nd of any security
requirement then
{
Design operator Nd for Fi to associate corresponding
security requirements to make it secure;
/*If any security requirement is mandatory requirement
for the functional requirement, Fi, to make it secure, then
we say that the security requirements Nd of the Fi
requirement*/
Call procedure Design-SM
Whether access is secure by using SM
{If weight is 1 then secure}
```

Figure 10. Design services procedure

Designing a system to be secure inevitably involves compromises. It is certainly possible to design multiple security measures into a system that will reduce the chances of a successful attack. However, security measures often require a lot of additional computation and so affect the overall performance of a system. For example, you can reduce the chances of confidential information being disclosed by encrypting that information. However, this means that users of the information have to wait for its decryption and this may slow down their work.

Our proposed methodology is based on Classical Water Fall Life Cycle Model, which provides a framework for software design methodologies. It is systemic way for system development; therefore it follows the principles of software engineering. The proposed methodology will help to begin the design task in a systematic and logical set of activities. The design of our proposed methodology is machine independent as well as application independent, however it will be a step towards automation.

Case Study: Information Retrieval System for Biological Data: Bioinformatics (or Computational Biology) involves the use of techniques including applied mathematics, informatics, statistics, computer science, artificial intelligence, chemistry and biochemistry to solve biological problems usually on the molecular level. Research in computational biology often overlaps with systems biology. Major research efforts in the field include sequence alignment, gene finding, genome assembly, protein structure alignment, protein structure prediction, prediction of gene expression and protein-protein interactions, and the modeling of evolution. The terms bioinformatics and computational biology are often used interchangeably.

The structure of biological molecules play a key role in determining their functions, but it is a difficult task to manage complex logical structure of biological data organization. Also, exponential growth of new biological data from the wet laboratories is contributing difficulties and complexity to the data management (such as data modeling, storage, retrieval and manipulation) and the software development methods for bioinformatics. The Bioinformatics systems have been taken as a case study for development in the cloud domain keeping in view their functional and peculiarly security requirements of such systems. To overcome the security issued of the databank, we propose in this paper a secured methodology.

In the system actual four different securities were introduced. However, some of the securities were repeated for different functional requirements and thus ten number of security were required in total by all functional requirements. After applying the Algebraic operators introduced in Analysis phase, actually only five security requirements out of ten were left to be implements for the system as shown in Figure 11 below .

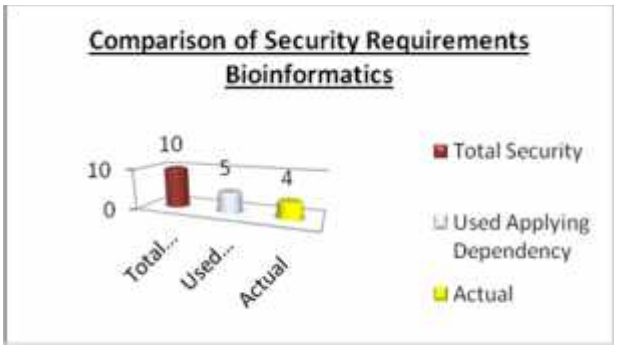

Figure 11: Comparison of Security Requirements for Bioinformatics

Figure 11, represents the comparison of security requirements and also depicts that after applying Dependency operator 50% saving in the securities. Which shows that a lot of security overhead has been decreased resultantly saving of cost and better performance of the system by using the proposed secured design methodology for bioinformatics application in the cloud domain.

Conclusion and future Direction: In this paper, we propose a design methodology for Cloud applications that overcomes the problems and deficiencies in the existing methodologies. Main features and objective of the proposed methodology is to develop secured applications in a systematically way. Time and cost benefits are the leading, motivating factors and measures for any business. The cloud provides a zero-investment method to quickly increase capacity or add capabilities. To fully utilize the potential of the cloud's, now the application developers are responsible to ensure that software is secure which they can ensure by considering security measures throughout the software development life cycle (SDLC). The existing methodologies do not effectively support the cloud paradigm user's security requirements in their each phase of development life cycle in an engineering fashion. In this work, we presented a systematic process for incorporating user's security concerns for cloud applications development. We expect that overall security mechanism will be significantly improved throughout the system development life-cycle instead of to be bolted upon top of the cloud software application. Resulting benefits includes but not limited to the saving of maintenance cost, less re-work and customers satisfaction regarding their security concerns. This work requires further research to propose security metrics for cloud application to have better security gains.

REFERENCES

Goble, C. and R. Stevens (2008). State of the nation in data integration for bioinformatics. J. Biomedical Informatics,41(5):687-693.

Insik K., Y. J. Jae, F. Todd, H. Tristan, P. Dennis (2012). "Cloud Computing for Comparative Genomics with Windows Azure Platform", Bioinform Online 8: 527–534.

Khan, M. U. G., S. Amanat, A. Nasir, , R. Iqbal and M. Idrees, (2015) A Computational Framework for Unified Biological Databank to Overcome Heterogeneity of Biological Data Format. *PAKISTAN ACADEMY OF SCIENCES*, 91.

Khan, P. I., M. U. G. Khan and A. Nasir (2015). A Unified Integration Model and Database Management System for Genomic Data. J. Faculty of Engineering and Technology, *22*(1): xx-xx.

Luscombe N. M., D. Greenbaum and M. Gerstein (2001)."What is bioinformatics? An introduction and overview", Yearbook of Medical Informatics 2001, (https://www.ebi.ac.uk/ luscombe/docs/imia_review.pdf).

Lynch, B. (2009). "Privacy in the Cloud Computing Era" A Microsoft Perspective, available athttp://download.microsoft.com.

Min-Huang, H., S. Yue, S.G. Ming, L. Kuang and M. Shyan (2003). A Unified, Adjustable, and Extractable Biological Data Mining-Broker, Springer. pp 773-777.

Nasir, A., A. Shah and M.U.G. Khan (2013). "Requirement Analysis Algebra: Designing Quality Software", J. Quality Technol. Manage. IX (II):107-136.

Rosenthal, A., P. Mork, M. H. Li, J. Stanford, D. Koester and P. Reynolds (2009)."Cloud computing: a new business paradigm for biomedical information sharing", J Biomed Inform: 43(2):342-53.